

METHOD AND DEVICE FOR SECONDARILY STORING EXPANDED NETWORK CONNECTION

Abstract of Reference

1

Publication number: JP2000122814

Publication date: 2000-04-28

Inventor: INOHARA SHIGEKAZU; NISHIZAWA ITARU;
WATANABE NAOKI; TOMITA AKI; MASHIERU
FUREDERIKO; ODAWARA HIROAKI; SAGAWA
NOBUTOSHI; SUGIE MAMORU

Applicant: HITACHI LTD

Classification:

- international: G06F15/16; G06F3/06; G06F12/00; G06F15/16;
G06F3/06; G06F12/00; (IPC1-7): G06F3/06; G06F12/00;
G06F15/16

- European:

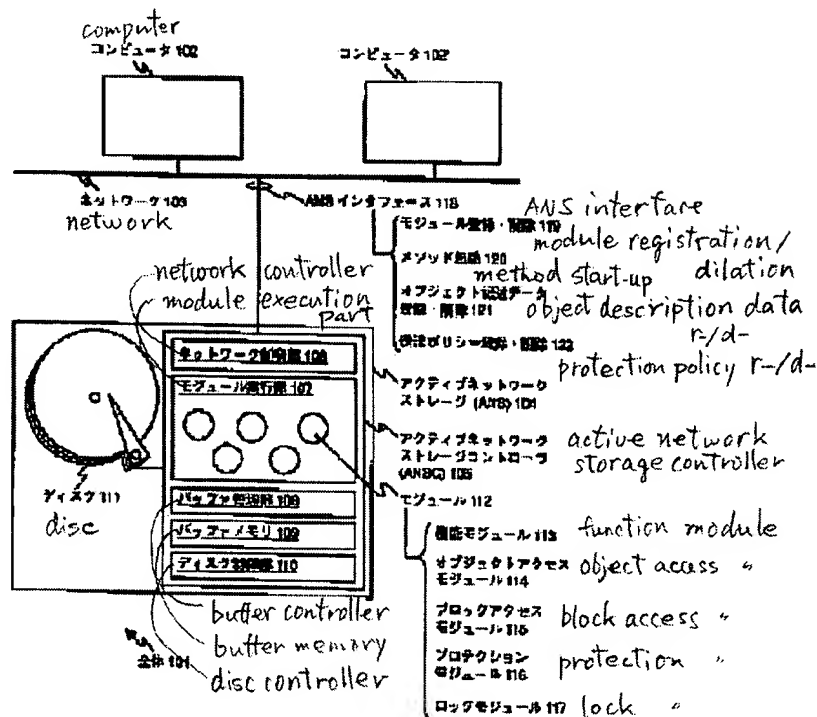
Application number: JP19980293277 19981015

Priority number(s): JP19980293277 19981015

Report a data error here

Abstract of JP2000122814

PROBLEM TO BE SOLVED: To suppress the development cost for a function and the total possession cost low when a storage having an effective function in a wide application range is continuously provided. **SOLUTION:** A storage is provided with an object access module 114 being a common part when high function input and output are realized to suppress the development cost for a module group realizing the function low, and the storage offers an interface 121 which registers and eliminates object description data in which a storing method to a secondary storage of an object is declaratively described. A module group is transferred to the storage via a management machine to suppress the total possession cost low. Thus, it is possible to provide a storage which continuously offers an effective function in a wide application range at low development cost and low total possession cost.



Data supplied from the esp@cenet database - Worldwide

【特許請求の範囲】

【請求項1】1つ以上の第1のコンピュータと、1つ以上の二次記憶装置と、該第1のコンピュータと該二次記憶装置とを接続するネットワークまたは入出力用信号線を備え、第1のコンピュータが1つ以上の応用プログラムを動作させ、該二次記憶装置が、電源断後もデータを保持することが可能な記憶媒体（二次記憶）を備え、該二次記憶が、複数の格納単位（ブロック）からなり、該二次記憶が、該応用プログラムが用いる1つ以上の応用向けデータ（オブジェクト）を1つ以上のブロックに格納しており、該二次記憶装置が、ブロック単位入出力の機能とオブジェクト単位入出力の機能とを第1のコンピュータに提供し、第1のコンピュータまたは第1のコンピュータと異なる第2のコンピュータから、ブロック単位入出力の機能を用いてオブジェクト単位入出力の機能を実現するプログラム・モジュール（オブジェクトアクセスモジュール）を受けた後、第1のコンピュータから該オブジェクトに対するオブジェクト単位入出力の要求を受け、該オブジェクトアクセスモジュールを実行することにより該要求によるオブジェクト単位入出力を行うことを特徴とする拡張型ネットワーク接続二次記憶方法。

【請求項2】前記オブジェクトアクセスモジュールが、オブジェクト、オブジェクト・オフセット、オブジェクト・オフセット・サイズ、オブジェクト・タグ（取り出すべきデータの種類の指定）、のいずれかの指定から、対応するデータの値または対応するデータのブロック内での位置を得る請求項1記載の拡張型ネットワーク接続二次記憶方法。

【請求項3】1つ以上の第1のコンピュータと、1つ以上の二次記憶装置と、該第1のコンピュータと該二次記憶装置とを接続するネットワークまたは入出力用信号線を備え、第1のコンピュータが1つ以上の応用プログラムを動作させ、該二次記憶装置が、電源断後もデータを保持することが可能な記憶媒体（二次記憶）を備え、該二次記憶が、複数の格納単位（ブロック）からなり、該二次記憶が、該応用プログラムが用いる1つ以上の応用向けデータ（オブジェクト）を1つ以上のブロックに格納しており、該二次記憶装置が、ブロック単位入出力の機能とオブジェクト単位入出力の機能とを第1のコンピュータに提供するコンピュータシステム内で動作する第1のコンピュータまたは第1のコンピュータと異なる第2のコンピュータとして動作し、ブロック単位入出力の機能を用いてオブジェクト単位入出力の機能を実現するプログラム・モジュール（オブジェクトアクセスモジュール）を、該二次記憶装置に登録または削除する拡張型ネットワーク接続二次記憶方法。

【請求項4】前記オブジェクトアクセスモジュールが、オブジェクト、オブジェクト・オフセット、オブジェクト・オフセット・サイズ、オブジェクト・タグ（取り出

すべきデータの種類の指定）、のいずれかの指定から、対応するデータの値または対応するデータのブロック内での位置を得る請求項3記載の第1のコンピュータ。

【請求項5】1つ以上の第1のコンピュータと、1つ以上の二次記憶装置と、該第1のコンピュータと該二次記憶装置とを接続するネットワークまたは入出力用信号線を備え、第1のコンピュータが1つ以上の応用プログラムを動作させ、該二次記憶装置が、電源断後もデータを保持することが可能な記憶媒体（二次記憶）を備え、該二次記憶が、複数の格納単位（ブロック）からなり、該二次記憶装置が、該応用プログラムが用いる1つ以上の応用向けデータ（オブジェクト）を1つ以上のブロックに格納しており、該二次記憶装置が、ブロック単位入出力の機能とオブジェクト単位入出力の機能とを第1のコンピュータに提供し、該機能を、第1のコンピュータまたは第1のコンピュータと異なる第2のコンピュータから、該二次記憶装置に送って実行することにより、ブロック単位入出力の機能を用いてオブジェクト単位入出力の機能を実現するオブジェクトアクセスモジュール。

【請求項6】オブジェクト、オブジェクト・オフセット、オブジェクト・オフセット・サイズ、オブジェクト・オブジェクト・タグ（取り出すべきデータの種類の指定）、のいずれかの指定から、対応するデータの値または対応するデータのブロック内での位置を得る請求項5記載のオブジェクトアクセスモジュール。

【請求項7】1つ以上の第1のコンピュータと、1つ以上の二次記憶装置と、該第1のコンピュータと該二次記憶装置とを接続するネットワークまたは入出力用信号線を備え、第1のコンピュータが1つ以上の応用プログラムを動作させ、該二次記憶装置が、電源断後もデータを保持することが可能な記憶媒体（二次記憶）を備え、該二次記憶が、複数の格納単位（ブロック）からなり、該二次記憶装置が、該応用プログラムが用いる1つ以上の応用向けデータ（オブジェクト）を1つ以上のブロックに格納しており、該二次記憶装置が、ブロック単位入出力の機能とオブジェクト単位入出力の機能とを第1のコンピュータに提供し、該オブジェクトが該二次記憶に格納されている方法を示すデータ（オブジェクト記述データ）を第1のコンピュータまたは第1のコンピュータと異なる第2のコンピュータから受けた後、第1のコンピュータから該オブジェクトに対するオブジェクト単位入出力の要求を受け、該オブジェクト記述データを用いて該オブジェクトの該二次記憶上の位置を特定することにより該要求の入出力を行う、ことを特徴とする拡張型ネットワーク接続二次記憶方法。

【請求項8】前記オブジェクト記述データが、オフセットとサイズによって、ブロック内データの属性またはブロック間参照関係を指定するデータである請求項7記載の拡張型ネットワーク接続二次記憶方法。

【請求項9】前記オブジェクト記述データが、構文解析

10

20

30

40

50

プログラム（パーザ）またはパーザ生成用文法によって、ブロック内データの属性またはブロック間参照関係を指定するデータである請求項7記載の拡張型ネットワーク接続二次記憶方法。

【請求項10】前記オブジェクト記述データが、1つ以上のブロックの特定部分に格納されているデータが、特定の値またはパターンであるかによって、前記オブジェクトのファイルフォーマットを特定するデータである請求項7記載の拡張型ネットワーク接続二次記憶方法。

【請求項11】1つ以上の第1のコンピュータと、1つ以上の二次記憶装置と、該第1のコンピュータと該二次記憶装置とを接続するネットワークまたは入出力用信号線を備え、第1のコンピュータが1つ以上の応用プログラムを動作させ、該二次記憶装置が、電源断後もデータを保持することが可能な記憶媒体（二次記憶）を備え、該二次記憶が、複数の格納単位（ブロック）からなり、該二次記憶が、該応用プログラムが用いる1つ以上の応用向けデータ（オブジェクト）を1つ以上のブロックに格納しており、該二次記憶装置が、ブロック単位入出力の機能とオブジェクト単位入出力の機能とを第1のコンピュータに提供するコンピュータシステム内で動作する第1のコンピュータまたは第1のコンピュータと異なる第2のコンピュータの機能を有し、該オブジェクトが該二次記憶に格納されている方法を示すデータ（オブジェクト記述データ）を該二次記憶装置に登録または削除することを特徴とするコンピュータ。

【請求項12】前記オブジェクト記述データが、オフセットとサイズによって、ブロック内のデータの並びまたはブロック間の参照関係を指定するデータである請求項11記載のコンピュータ。

【請求項13】前記オブジェクト記述データが、構文解析プログラム（パーザ）またはパーザ生成用文法によって、ブロック内のデータの並びまたはブロック間の参照関係を指定するデータである請求項11に記載のコンピュータ。

【請求項14】前記オブジェクト記述データが、1つ以上のブロックの特定の部分に格納されているデータが、特定の値またはパターンであるかによって、前記オブジェクトのファイルフォーマットを特定するデータである請求項11に記載のコンピュータ。

【請求項15】前記オブジェクトアクセスモジュールが、請求項7乃至14記載のオブジェクト記述データからオブジェクトが二次記憶に格納されている方法を得る拡張型ネットワーク接続二次記憶方法。

【請求項16】前記オブジェクトアクセスモジュールが、請求項7乃至請求項14記載のオブジェクト記述データからオブジェクトが二次記憶に格納されている方法を得る請求項3記載の第1のコンピュータ。

【請求項17】請求項7乃至請求項14に記載のオブジェクト記述データからオブジェクトが二次記憶に格納さ

れている方法を得る請求項5記載のオブジェクトアクセスモジュール。

【請求項18】1つ以上の第1のコンピュータと、1つ以上の二次記憶装置と、該第1のコンピュータと該二次記憶装置とを接続するネットワークまたは入出力用信号線を備え、第1のコンピュータが1つ以上の応用プログラムを動作させ、該二次記憶装置が、電源断後もデータを保持することが可能な記憶媒体（二次記憶）を備え、該二次記憶が、複数の格納単位（ブロック）からなり、該二次記憶が、該応用プログラムが用いる1つ以上の応用向けデータ（オブジェクト）を1つ以上のブロックに格納しており、該二次記憶装置が、ブロック単位入出力の機能と応用プログラム向けの入出力（高機能入出力）の機能とを第1のコンピュータに提供し、コンピュータシステムの二次記憶装置であって、ブロック単位入出力の機能を用いてオブジェクト単位入出力の機能を実現するオブジェクトアクセスモジュールを保持し、該オブジェクトアクセスモジュールを用いて該高機能入出力を実現するモジュール（機能モジュール）を、第1のコンピュータまたは第1のコンピュータと異なる第2のコンピュータから受けた後に、第1のコンピュータから該高機能入出力の要求を受け、該機能モジュールを実行することにより該要求の入出力を行うことを特徴とする拡張型ネットワーク接続二次記憶方法。

【請求項19】1つ以上の第1のコンピュータと、1つ以上の二次記憶装置と、該第1のコンピュータと該二次記憶装置とを接続するネットワークまたは入出力用信号線を備え、第1のコンピュータが1つ以上の応用プログラムを動作させ、該二次記憶装置が、電源断後もデータを保持することが可能な記憶媒体（二次記憶）を備え、該二次記憶が、複数の格納単位（ブロック）からなり、該二次記憶が、該応用プログラムが用いる1つ以上の応用向けデータ（オブジェクト）を1つ以上のブロックに格納しており、該二次記憶装置が、ブロック単位入出力の機能と、該応用プログラム向けの入出力（高機能入出力）の機能と、を第1のコンピュータに提供するコンピュータシステム内で動作する第1のコンピュータまたは第1のコンピュータと異なる第2のコンピュータとして動作し、該二次記憶装置が、オブジェクト単位入出力の機能をブロック単位入出力の機能を用いて実現するモジュール（オブジェクトアクセスモジュール）を保持し、該オブジェクトアクセスモジュールを利用して該高機能入出力を実現するモジュール（機能モジュール）を、該コンピュータに登録または削除することを特徴とするコンピュータ。

【請求項20】1つ以上の第1のコンピュータと、1つ以上の二次記憶装置と、該第1のコンピュータと該二次記憶装置とを接続するネットワークまたは入出力用信号線を備え、第1のコンピュータが1つ以上の応用プログラムを動作させ、該二次記憶装置が、電源断後もデータ

10

20

30

40

50

を保持することが可能な記憶媒体（二次記憶）を備え、該二次記憶が、複数の格納単位（ブロック）からなり、該二次記憶が、該応用プログラムが用いる1つ以上の応用向けデータ（オブジェクト）を1つ以上のブロックに格納しており、該二次記憶装置が、ブロック単位入出力の機能と応用プログラム向けの入出力（高機能入出力）の機能とを第1のコンピュータに提供し、第1のコンピュータまたは第1のコンピュータと異なる第2のコンピュータから、該二次記憶装置に送られ、該二次記憶装置で実行され、ブロック単位入出力の機能を用いてオブジェクト単位入出力の機能を実現するモジュール（オブジェクトアクセスモジュール）を利用して該高機能入出力を実現することを特徴とするプログラム・モジュール。

【請求項21】1つ以上の第1のコンピュータと、1つ以上の二次記憶装置と、該第1のコンピュータと該二次記憶装置とを接続するネットワークまたは入出力用信号線を備え、第1のコンピュータが1つ以上の応用プログラムを動作させ、該二次記憶装置が、電源断後もデータを保持することが可能な記憶媒体（二次記憶）を備え、該二次記憶が、複数の格納単位（ブロック）からなり、該二次記憶が、該応用プログラムが用いる1つ以上の応用向けデータ（オブジェクト）を1つ以上のブロックに格納しており、該二次記憶装置が、ブロック単位入出力の機能と応用プログラム向けの入出力（高機能入出力）の機能とを第1のコンピュータに提供し、該二次記憶装置が、ブロック単位入出力の機能を用いてオブジェクト単位入出力の機能を実現するモジュール（オブジェクトアクセスモジュール）を保持し、該二次記憶装置が、該オブジェクトアクセスモジュールを利用して該高機能入出力を実現するモジュール（機能モジュール）を保持し、該機能モジュールがオブジェクトアクセスモジュールのメソッドを起動しようとした際に、該メソッド起動の可否を判定する、ことを特徴とする二次記憶装置及び該二次記憶装置上のプロテクションモジュール。

【請求項22】前記プロテクションモジュールが、前記コンピュータシステム内で動作する第1のコンピュータまたは第1のコンピュータと異なる第2のコンピュータによって、前記二次記憶装置へ登録、または前記二次記憶装置から削除される請求項21記載のプロテクションモジュール。

【請求項23】1つ以上の第1のコンピュータと、1つ以上の二次記憶装置と、該第1のコンピュータと該二次記憶装置とを接続するネットワークまたは入出力用信号線を備え、第1のコンピュータが1つ以上の応用プログラムを動作させ、該二次記憶装置が、電源断後もデータを保持することが可能な記憶媒体（二次記憶）を備え、該二次記憶が、複数の格納単位（ブロック）からなり、該二次記憶が、該応用プログラムが用いる1つ以上の応用向けデータ（オブジェクト）を1つ以上のブロックに格納しており、該二次記憶装置が、ブロック単位入出力

の機能と応用プログラム向けの入出力（高機能入出力）の機能とを第1のコンピュータに提供し、該二次記憶装置が、ブロック単位入出力の機能を用いてオブジェクト単位入出力の機能を実現するオブジェクトアクセスモジュール（機能モジュール）を保持し、該二次記憶装置が、該オブジェクトアクセスモジュールを利用して該高機能入出力を実現するモジュール（機能モジュール）を保持し、該機能モジュールがオブジェクトアクセスモジュールのメソッドを起動しようとした際に、該メソッド起動の可否を判定するモジュール（プロテクションモジュール）を、第1のコンピュータまたは第1のコンピュータと異なる第2のコンピュータから受ける、ことを特徴とする拡張型ネットワーク接続二次記憶方法。

【請求項24】1つ以上の第1のコンピュータと、1つ以上の二次記憶装置と、該第1のコンピュータと該二次記憶装置とを接続するネットワークまたは入出力用信号線を備え、第1のコンピュータが1つ以上の応用プログラムを動作させ、該二次記憶装置が、電源断後もデータを保持することが可能な記憶媒体（二次記憶）を備え、該二次記憶が、複数の格納単位（ブロック）からなり、該二次記憶が、該応用プログラムが用いる1つ以上の応用向けデータ（オブジェクト）を1つ以上のブロックに格納しており、該二次記憶装置が、ブロック単位入出力の機能と、該応用プログラム向けの入出力（高機能入出力）の機能と、を第1のコンピュータに提供するコンピュータシステム内で動作する第1のコンピュータまたは第1のコンピュータと異なる第2のコンピュータとして動作し、該二次記憶装置が、ブロック単位入出力の機能を用いてオブジェクト単位入出力の機能を実現するオブジェクトアクセスモジュールを保持し、該二次記憶装置が、該オブジェクトアクセスモジュールを利用して該高機能入出力を実現するモジュール（機能モジュール）を保持し、該機能モジュールがオブジェクトアクセスモジュールのメソッドを起動しようとした際に、該メソッド起動の可否を判定するプロテクションモジュールを該二次記憶装置に登録または削除することを特徴とするコンピュータ。

【請求項25】1つ以上の第1のコンピュータと、1つ以上の二次記憶装置と、該第1のコンピュータと該二次記憶装置とを接続するネットワークまたは入出力用信号線を備え、第1のコンピュータが1つ以上の応用プログラムを動作させ、該二次記憶装置が、電源断後もデータを保持することが可能な記憶媒体（二次記憶）を備え、該二次記憶が、複数の格納単位（ブロック）からなり、該二次記憶が、該応用プログラムが用いる1つ以上の応用向けデータ（オブジェクト）を1つ以上のブロックに格納しており、該二次記憶装置が、ブロック単位入出力の機能と応用プログラム向けの入出力（高機能入出力）の機能とを第1のコンピュータに提供し、該二次記憶装置が、ブロック単位入出力の機能を用いてオブジェクト

単位入出力の機能を実現するオブジェクトアクセスモジュールを保持し、該オブジェクトアクセスモジュールが包含関係のある複数のオブジェクトを外部に提供する際に、該複数のオブジェクトの包含関係を考慮した排他制御の機能を外部に提供する、ことを特徴とする二次記憶装置上のロックモジュール。

【請求項 26】前記ロックモジュールは、前記第 1 のコンピュータまたは第 1 のコンピュータと異なる第 2 のコンピュータによって、前記二次記憶装置へ登録、または前記二次記憶装置から削除される請求項 25 記載のロックモジュール。

【請求項 27】1 つ以上の第 1 のコンピュータと、1 つ以上の二次記憶装置と、該第 1 のコンピュータと該二次記憶装置とを接続するネットワークまたは入出力用信号線を備え、第 1 のコンピュータが 1 つ以上の応用プログラムを動作させ、該二次記憶装置が、電源断後もデータを保持することが可能な記憶媒体（二次記憶）を備え、該二次記憶が、複数の格納単位（ブロック）からなり、該二次記憶が、該応用プログラムが用いる 1 つ以上の応用向けデータ（オブジェクト）を 1 つ以上のブロックに格納しており、該二次記憶装置が、ブロック単位入出力の機能と、該応用プログラム向けの入出力（高機能入出力）の機能と、を第 1 のコンピュータに提供するコンピュータシステムの二次記憶装置であって、該二次記憶装置が、ブロック単位入出力の機能を用いてオブジェクト単位入出力の機能を実現するオブジェクトアクセスモジュールを保持し、該オブジェクトアクセスモジュールが包含関係のある複数のオブジェクトを外部に提供する際に、該複数のオブジェクトの包含関係を考慮した排他制御の機能を外部に提供するロックモジュールを、第 1 のコンピュータまたは第 1 のコンピュータと異なる第 2 のコンピュータから受ける、ことを特徴とする拡張型ネットワーク接続二次記憶方法。

【請求項 28】1 つ以上の第 1 のコンピュータと、1 つ以上の二次記憶装置と、該第 1 のコンピュータと該二次記憶装置とを接続するネットワークまたは入出力用信号線を備え、第 1 のコンピュータが 1 つ以上の応用プログラムを動作させ、該二次記憶装置が、電源断後もデータを保持することが可能な記憶媒体（二次記憶）を備え、該二次記憶が、複数の格納単位（ブロック）からなり、該二次記憶が、該応用プログラムが用いる 1 つ以上の応用向けデータ（オブジェクト）を 1 つ以上のブロックに格納しており、該二次記憶装置が、ブロック単位入出力の機能と、該応用プログラム向けの入出力（高機能入出力）の機能と、を第 1 のコンピュータに提供するコンピュータシステム内で動作する第 1 のコンピュータまたは第 1 のコンピュータと異なる第 2 のコンピュータとして動作し、該二次記憶装置が、ブロック単位入出力の機能を用いてオブジェクト単位入出力の機能を実現するオブジェクトアクセスモジュールを保持し、該オブジェクト

アクセスモジュールが包含関係のある複数のオブジェクトを外部に提供する際に、該複数のオブジェクトの包含関係を考慮した排他制御の機能を外部に提供するロックモジュールを、該二次記憶装置に登録または二次記憶装置から削除する、ことを特徴とするコンピュータ。

【請求項 29】1 つ以上の第 1 のコンピュータと、1 つ以上の二次記憶装置と、1 つの第 2 のコンピュータ（管理マシン）と、該第 1 のコンピュータと該二次記憶装置と該管理マシンを接続するネットワークまたは入出力用信号線を備え、第 1 のコンピュータが 1 つ以上の応用プログラムを動作させ、第 2 のコンピュータが該二次記憶装置のリストを保持しており、該二次記憶装置が、電源断後もデータを保持することが可能な記憶媒体（二次記憶）を備え、該二次記憶が、複数の格納単位（ブロック）からなり、該二次記憶装置が、ブロック単位入出力の機能と、該応用プログラム向けの入出力（高機能入出力）の機能またはオブジェクト単位入出力の機能とを第 1 のコンピュータに提供し、該高機能入出力を実現するプログラム・モジュール（モジュール）を第 1 のコンピュータが第 2 のコンピュータに送り、第 2 のコンピュータが、該モジュールを受け取り、該リストに保持された該二次記憶装置の一部または全部に該モジュールを送り、該二次記憶装置が該モジュールを受け取り、第 1 のコンピュータが該高機能入出力の要求を二次記憶装置に送り、該二次記憶装置が該モジュールを起動することにより該高機能入出力を行う、ことを特徴とする管理マシン。

【請求項 30】前記プログラム・モジュールを前記二次記憶装置向けにコンパイルするコンパイラを備え、第 1 のコンピュータから受け取った前記モジュールを、該コンパイラによってコンパイルし、該二次記憶装置の一部または全部にコンパイル済みモジュールを送る請求項 29 記載の管理マシン。

【請求項 31】前記二次記憶装置の機種データを保持し、前記二次記憶装置の機種毎に、モジュールをコンパイルする 1 つ以上のコンパイラを備え、第 1 のコンピュータから受け取った前記モジュールを、該 1 つ以上のコンパイラによって送付先の二次記憶装置向けにコンパイルし、該二次記憶装置の一部または全部にコンパイル済みモジュールを送る請求項 30 記載の管理マシン。

【請求項 32】1 つ以上の第 1 のコンピュータと、1 つ以上の二次記憶装置と、1 つの第 2 のコンピュータ（管理マシン）と、該第 1 のコンピュータと該二次記憶装置と該管理マシンを接続するネットワークまたは入出力用信号線を備え、第 1 のコンピュータが 1 つ以上の応用プログラムを動作させ、第 2 のコンピュータが該二次記憶装置のリストを保持しており、該二次記憶装置が、電源断後もデータを保持することが可能な記憶媒体（二次記憶）を備え、該二次記憶が、複数の格納単位（ブロック）からなり、該二次記憶装置が、ブロック単位入出力

の機能と、該応用プログラム向けの入出力（高機能入出力）の機能またはオブジェクト単位入出力の機能とを第1のコンピュータに提供し、前記高機能入出力を実現するプログラム・モジュール（モジュール）を第1のコンピュータが第2のコンピュータに送り、第2のコンピュータが、該モジュールを受け取り、該リストに保持された前記二次記憶装置の一部または全部に該モジュールを送り、該二次記憶装置が該モジュールを受け取り、第1のコンピュータが該高機能入出力の要求を二次記憶装置に送り、該二次記憶装置が該モジュールを実行することにより該高機能入出力を行うことを特徴とする拡張型ネットワーク接続二次記憶方法。

【請求項33】1つ以上の第1のコンピュータと、1つ以上の二次記憶装置と、該第1のコンピュータと該二次記憶装置とを接続するネットワークまたは入出力用信号線を備え、第1のコンピュータが1つ以上の応用プログラムを動作させ、該二次記憶装置が、電源断後もデータを保持することが可能な記憶媒体（二次記憶）を備え、該二次記憶が、複数の格納単位（ブロック）からなり、該二次記憶が、該応用プログラムが用いる1つ以上の応用向けデータ（オブジェクト）を1つ以上のブロックに格納しており、該二次記憶装置が、ブロック単位入出力の機能と、該応用プログラム向けの入出力（高機能入出力）の機能またはオブジェクト単位入出力の機能とを第1のコンピュータに提供し、第1のコンピュータまたは第1のコンピュータと異なる第2のコンピュータから、ブロック単位入出力の機能を用いてオブジェクト単位入出力の機能を実現するプログラム・モジュール（オブジェクトアクセスモジュール）または該高機能入出力を実現するプログラム・モジュールを受け、該二次記憶装置が、該モジュールを高速実行が可能な実行プログラムへコンパイルするコンパイラを備え、該二次記憶装置が該コンパイラで該モジュールをコンパイルし、第1のコンピュータから該オブジェクトに対するオブジェクト単位入出力の要求または高機能入出力の要求を受け、該コンパイルしたモジュールを実行することにより該要求の入出力を行う、ことを特徴とする拡張型ネットワーク接続二次記憶方法。

【請求項34】1つ以上の第1のコンピュータと、1つ以上の二次記憶装置と、該第1のコンピュータと該二次記憶装置とを接続するネットワークまたは入出力用信号線を備え、第1のコンピュータが1つ以上の応用プログラムを動作させ、該二次記憶装置が、電源断後もデータを保持することが可能な記憶媒体（二次記憶）を備え、該二次記憶が、複数の格納単位（ブロック）からなり、該二次記憶が、該応用プログラムが用いる1つ以上の応用向けデータ（オブジェクト）を1つ以上のブロックに格納しており、該二次記憶装置が、ブロック単位入出力の機能と応用プログラム向けの入出力（高機能入出力）の機能とを第1のコンピュータに提供し、該高機能入出

力の機能として、第1のコンピュータの要求に応じて1つのオブジェクトの複数の部分と、該部分を格納する場所である二次記憶装置との対応関係（ストライプ構成情報）を返す機能を備える、ことを特徴とする拡張型ネットワーク接続二次記憶方法。

【請求項35】1つ以上の第1のコンピュータと、1つ以上の二次記憶装置と、該第1のコンピュータと該二次記憶装置とを接続するネットワークまたは入出力用信号線を備え、第1のコンピュータが1つ以上の応用プログラムを動作させ、該二次記憶装置が、電源断後もデータを保持することが可能な記憶媒体（二次記憶）を備え、該二次記憶が、複数の格納単位（ブロック）からなり、該二次記憶が、該応用プログラムが用いる1つ以上の応用向けデータ（オブジェクト）を1つ以上のブロックに格納しており、ブロック単位入出力の機能とオブジェクト単位入出力の機能とを第1のコンピュータに提供する手段と、第1のコンピュータまたは第1のコンピュータと異なる第2のコンピュータから、ブロック単位入出力の機能を用いてオブジェクト単位入出力の機能を実現するプログラム・モジュール（オブジェクトアクセスモジュール）を受け取る手段と、該オブジェクトアクセスモジュールを受けた後、第1のコンピュータから該オブジェクトアクセスモジュールに対するオブジェクト単位入出力の要求を受け取る手段と、該オブジェクトアクセスモジュールを実行する手段と、を有することを特徴とする拡張型ネットワーク接続二次記憶装置。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】本発明はコンピュータシステムに関し、特に応用プログラムに適した新たな機能を追加・拡張可能な二次記憶方法及び装置に関する。

【0002】

【従来の技術】今日のコンピュータシステムの主要な構成要素は、コンピュータ（プロセッサとメモリ、周辺機器からなる）とネットワークと二次記憶装置（ストレージ）である。これまでストレージは、コンピュータに付属する装置として存在することが多かったが、近年、このような状況が変化しつつある。

【0003】第1に、ネットワークの普及により、複数のコンピュータがストレージを共有する機会が増えた。他のコンピュータからのネットワーク経由のストレージ入出力が、ストレージを接続したコンピュータのプロセッサ処理能力がボトルネックとなって滞る事態を招いている。第2に、ストレージ容量およびストレージに対する要求スループットは、年々増大している。「データウェアハウス用途のストレージ容量の要求は、9ヶ月で2倍になる」という予測（Greg's Law）もある。このため、1つのコンピュータに接続するストレージ数が増大し、やはりコンピュータがストレージ入出力のボトルネックとなる恐れが出てきている。第3に、ハ

ードディスク制御用LSIの高集積化の進行によって、ストレージの高機能化の可能性が増大している。

【0004】これらの背景から、ストレージの制御用LSIに新たな機能を付加することが考えられている。新たな機能の候補は、ネットワークインタフェースと、応用プログラム向けの高度機能である。

【0005】ネットワークインタフェースをストレージが備えることにより、ストレージをネットワークに直結することができる。これによりストレージは、複数のコンピュータからの入出力要求を、1つのコンピュータを介することなく、直接受けることができる。

【0006】現在、ストレージとコンピュータとの間の最も代表的なインタフェースはブロック入出力であるが、これよりも応用毎の高度機能（例えば、ソーティング、画像処理、データベースシステムにおける基本演算（例えば、選択処理、写像処理、結合処理、集計処理等）をストレージが備えることにより、コンピュータのプロセッサ処理の一部をストレージが受け持つことが可能となる。

【0007】ネットワークインタフェースを備え、かつファイルシステムの一部の機能を備えたストレージの提案の代表例としては、Garth A. Gibson 他著の文献“A Cost-Effective, High-Bandwidth Storage Architecture” (Proceedings of the 8th Conference on Architectural Support for Programming Languages and Operating Systems, 1998年ACM発行; 以後文献1と記す) と Steven R. Soltis 他著の文献“The Global File System” (Proceedings of the Fifth NASA Goddard Space Flight Center Conference on Mass Storage Systems and Technologies, 1996年NASA発行; 以後文献2と記す) に記載のシステムがある。

【0008】また複数の応用を想定して、ストレージを高機能化する提案の代表例としては、Erik Riedel 他著の文献“Active Disks: Remote Execution for Network-Attached Storage” (Technical Report CMU-CS-97-198, 1997年Carnegie Mellon University発行; 以後文献3と記す)、Anurag Acharya 他著の文献“Active Disks: Programming Model, Algorithms and Evaluation” (Proceedings of the 8th Conference on Architectural Sup

port for Programming Languages and Operating Systems, 1998年ACM発行; 以後文献4と記す)、Kimberly Keeton 他著の文献“A Case for Intelligent Disks (IDISs)” (SIGMOD Record, Volume 27, Number 3, 1998年ACM発行; 以後文献5と記す) に記載のシステムがある。

【0009】文献3、文献4、文献5はいずれも、ネットワーク経由で高機能を実現するプログラム・モジュールをコンピュータからストレージに転送（ダウンロード）することについて言及している。このようなモジュールを記述する言語の案としては、J. Gosling 他著の文献“The Java Language Specification” (1996年Addison-Wesley発行; 以後文献6と記す) に記載の言語がある。

【0010】図2を用いて、従来のストレージの代表であるサーバ接続型ストレージ (Server-Attached Disk; SAD) の構成を説明する。

【0011】1つ以上のSAD型ストレージ203は、入出力用信号線202によって通常1つのコンピュータ201と接続される。場合によっては複数のコンピュータと接続されることもある。SAD型ストレージ203はストレージコントローラ204とディスク209からなり、ストレージコントローラ204はさらにインタフェース制御部205、バッファ管理部206、バッファメモリ207、ディスク制御部208からなる。

【0012】ディスク209は電源断後もデータを保持することが可能な記憶媒体（二次記憶）である。インタフェース制御部205は、外部から入出力用信号線202経由で送られてくる入出力要求やその他の通信を受け、また入出力要求への応答やその他の通信を入出力用信号線202へ送り出す。バッファ管理部206は、バッファメモリ207の制御を行う部分である。バッファメモリ207は、ディスク209から得たデータを一時的にためておくメモリである。ディスク制御部208は、ディスク209にブロックの読み出し・書き込みを行わせる制御を行う。

【0013】SAD型ストレージ203とコンピュータ201の間のインタフェースであるSADインタフェース210は、ブロック単位入出力を提供する。

【0014】図3を用いて、近年登場しつつあるネットワーク接続型ストレージ (Network-Attached Storage NAS) の構成を説明する。

【0015】1つ以上のNAS型ストレージ303は、ネットワーク302によって1つ以上のコンピュータ301、301'、…と接続される。NAS型ストレージ303はストレージコントローラ304とディスク309からなり、ストレージコントローラ304はさらにネ

ネットワーク制御部305、バッファ管理部306、バッファメモリ307、ディスク制御部308からなる。

【0016】ネットワーク制御部305は、外部からネットワーク302経由で送られてくる入出力要求やその他の通信を受け、また入出力要求への応答やその他の通信をネットワーク302へ送り出す。ディスク309、バッファ管理部306、バッファメモリ307、ディスク制御部308は、それぞれディスク209、バッファ管理部206、バッファメモリ207、ディスク制御部208と同等の機能を持つ。

【0017】NAS型ストレージ303とコンピュータ301、301'、…の間のインタフェースであるNASインタフェース310は、ブロック単位入出力を提供する。

【0018】図4を用いて、従来のSAD型ストレージを拡張した高機能SAD型ストレージの構成を説明する。

【0019】1つ以上の高機能SAD型ストレージ403は、入出力用信号線402によって通常1つのコンピュータ401と接続される。場合によっては複数のコンピュータと接続されることもある。高機能SAD型ストレージ403はストレージコントローラ404とディスク409からなり、ストレージコントローラ404はさらにインタフェース制御部405、バッファ管理部406、バッファメモリ407、ディスク制御部408、及び、高機能処理部411からなる。

【0020】入出力用信号線402、インタフェース制御部405、バッファ管理部406、バッファメモリ407、ディスク制御部408、ディスク409、はそれぞれ、入出力用信号線202、インタフェース制御部205、バッファ管理部206、バッファメモリ207、ディスク制御部208、ディスク209と同等の機能を提供する。高機能処理部411は、特定の応用向けの高機能、例えばソーティング、画像処理、データベースシステムの基本演算、例えば選択処理、写像処理、結合処理、集計処理等を提供する。高機能SADインタフェース410は、ブロック単位入出力とともに、高機能処理部411が提供する高度処理を利用するためのインタフェースを持つ。

【0021】図5を用いて、近年提案されている高機能NAS型ストレージの構成を説明する。

【0022】1つ以上の高機能NAS型ストレージ503は、ネットワーク502によって1つ以上のコンピュータ501、501'、…と接続される。高機能NAS型ストレージ503はストレージコントローラ504とディスク509からなり、ストレージコントローラ504はさらにネットワーク制御部505、バッファ管理部506、バッファメモリ507、ディスク制御部508、及び高機能処理部411からなる。

【0023】ネットワーク制御部505、ディスク50

9、バッファ管理部506、バッファメモリ507、ディスク制御部508は、それぞれネットワーク制御部305、ディスク309、バッファ管理部306、バッファメモリ307、ディスク制御部308と同等の機能を持つ。

【0024】高機能処理部411は、特定の応用向けの高機能、例えばソーティング、画像処理、データベースシステムの基本演算（例えば、選択処理、写像処理、結合処理、集計処理等）を提供する。高機能NASインタフェース510は、ブロック単位入出力とともに、高機能処理部411が提供する高度処理を利用するためのインタフェースを持つ。また、文献4に記載のシステム及び文献5に記載のシステムでは、高機能処理部411が備える機能を外部からダウンロードすることができる。

【0025】

【発明が解決しようとする課題】ネットワークに直接結合し、または高機能を提供するストレージを実現する際に、もっとも根本的な課題は、広い応用範囲に対して有効な機能を継続的に提供できるストレージにすることである。応用範囲の広さは、市場の広さにつながり、市場の広さは開発コストの低減と開発スピードの向上につながる。過去に多くのデータベース専用マシンが提案され、ことごとく失敗したが、その原因は応用範囲の狭さにあったと考えられている。応用範囲が狭かったために十分な開発コストがかけられず、そのために汎用品を用いた製品の競争力（開発速度、速度向上等）について行けなかったのである。

【0026】広い応用範囲に対して有効な機能を継続的に提供するストレージを実現するためには、機能の拡張性が高いこと、機能の開発コストが低いこと、及び総所有コスト（Total Cost of Ownership）の観点からコストパフォーマンスが高いこと、の3点が要件である。最も大きくは、これら3点が本発明が解決しようとする課題である。

【0027】従来技術では、応用毎の高機能が必要に応じてストレージにダウンロードすることで拡張性を考慮しているが、開発コストと総所有コストの要件が十分考慮されていない。

【0028】文献1に記載のシステムと文献2に記載のシステムでは、ブロックアクセスの上にファイルシステムを作りつけにしておき、このため応用範囲が限られている。文献3に記載のシステムは、複数の高機能を備えることを考察しているが、これら複数の高機能をどのように作るかについては不明である。文献4は、従来のOSに近いソフトウェア階層の上に複数の高機能を実現する提案である。しかし、応用毎に従来のOS部分にあたるソフトウェアの作りも異なる。例えばRDBMSはOSのファイルシステムを使わないので、RDBMSにとってはファイルシステムは不要である。すなわち

従来のソフトウェア階層をそのままストレージに持ち込んでも、ストレージが担うべき広い応用範囲に対応することが難しい。文献5に記載のシステムはまだ構想段階であるが、RDBMSに焦点をあてており、応用範囲を限定している。

【0029】従来技術で十分考慮されていない開発コストと総所有コストの要件についてさらに考察する。

【0030】開発コストに関しては、応用毎の高度機能を一一つ別個に開発していたのでは、開発コストが増大し、競争力の低下につながる。応用毎の高度機能を実現するプログラム・モジュール（以下モジュールと記す）を低開発コストで開発する必要がある。高度機能を実現するモジュールは応用毎に千差万別であるが、これら高度機能の共通部分をくりだせれば、共通部分を重複して開発する必要がなくなるとともに共通部分のデバッグも不要になり、低開発コストを実現できる。また、開発したモジュールが高速に実行できる機構が提供できれば、チューニングに要する開発時間が短縮できるため、開発コストが低減できる。

【0031】すなわち、開発コストの要件に関しては、以下の課題がある。

【0032】(A) 複数の応用向け高度機能の共通部分をストレージが提供する。

【0033】(B) 該共通部分を低開発コストで実現する。

【0034】(C) 該共通部分を用いて複数の応用向け高度機能を実現する。

【0035】(D) 複数の応用向け高度機能が共通部分を使う際に保護を行う。

【0036】(E) 前記応用向け高度機能が共通部分を使う際に排他制御を行う。

【0037】(F) モジュールを高速に実行できる機構を提供する。

【0038】さらに、総所有コストの要件に関しては、ストレージがネットワーク上に多数存在し得ることを鑑みる必要性から、次の課題がある。

【0039】(G) 多数のストレージが存在する際に、モジュールを配布する。

【0040】がある。

【0041】従って、本発明の目的は、以上(A)から(G)の各課題を解決することにより、広い応用範囲に対して有効な機能を継続的に提供し、次世代のストレージの実現を可能にした拡張型ネットワーク接続二次記憶方法及び装置を提供することである。

【0042】

【課題を解決するための手段】以下に上記(A)から(G)の課題を解決する手段(a)から(g)を順に述べる。

【0043】(a) 本発明では、複数の応用の高度機能の共通部分として、応用が用いるデータの塊（オブジェ

クト）に着目した。オブジェクトの例は、データベースシステムのテーブル、レコード、カラム等のデータ、ファイルシステムではファイル、また、ファイルシステム上の特定のファイルフォーマットを扱う応用ではその特定のファイルフォーマットのファイル、などである。オブジェクトのデータを二次記憶上に配置する方法は、通常、応用または応用分野に対して決まっており、応用毎の高度機能の数に比べずっと少ない。このため、二次記憶上に格納されたオブジェクトにアクセスするモジュール（以下オブジェクトアクセスモジュールと呼ぶ）を高度機能を実現するモジュール（以下機能モジュールと呼ぶ）と分離して、これら2種のモジュール間のインタフェースを明確に定義する。

【0044】その上で、様々な応用分野に対応するため、オブジェクトアクセスモジュールを、コンピュータからストレージにダウンロード可能とすることにより、ストレージの拡張性を提供する。オブジェクトアクセスモジュールは、オブジェクトが二次記憶にどのように格納されているかを知っており、このモジュールによって応用毎のオブジェクトを二次記憶の1つ以上のブロックから取り出すことを可能とする。これらにより、様々な応用向けの機能モジュールの開発コストを削減することが可能となる。

【0045】なお、本発明が対象とするオブジェクトは、単に連続ブロックに存在する可変長のデータなどのような単純な構造を持つものではない。オブジェクトは、一般に複数の非連続なブロックにまたがって存在し、ブロック間の参照関係も存在する。例えば、Ureshi Vahalia 著の文献"UNIX Internals: The New Frontiers" 262頁 266頁、1996年Prentice-Hall発行；以後文献7と記す）に記載のファイルシステムでは、ひとつのファイルが1段、2段または3段のブロック間参照によって繋がっている。また、データベースのインデックスとして頻繁に用いられるB-treeやHashテーブルも、多段のブロック間参照を含む複雑なオブジェクトである。

【0046】(b) オブジェクトが二次記憶に格納されている方法（どのブロックのどの部分にどのような順序で格納されているか）のうち宣言的に書ける部分は、宣言的に書くことによって開発コストを削減できる。このため、本発明は、オブジェクトが二次記憶に格納されている方法を宣言的に記述したオブジェクト記述データを導入する。

【0047】オブジェクト記述データを、コンピュータからストレージにダウンロード可能とすることにより、ストレージの拡張性を提供しつつ、オブジェクト単位のアクセスを低開発コストで実現することを可能とする。

【0048】(c) 本発明では、機能モジュールを、オブジェクトアクセスモジュールの機能を用いて実現する

ことによって、開発コストを低減することを可能とする。機能モジュールをコンピュータからストレージにダウンロード可能とすることにより、ストレージの拡張性を提供しつつ、応用毎の高度機能を実現する。

【0049】(d) オブジェクトアクセスモジュールは共用のモジュールなので、安定したモジュールになりやすい。一方、機能モジュールは応用ごと、高度機能毎のモジュールなので、使用頻度も少なく、バグが取れて安定するのに時間がかかる。本発明はこの点に着目し、機能モジュールからのオブジェクトアクセスモジュールの10 利用を保護するモジュール(プロテクションモジュール)を導入する。

【0050】また、プロテクションモジュールをコンピュータからストレージにダウンロード可能とすることで、保護の方法をオブジェクト毎や応用プログラム毎の拡張を可能とする。

【0051】(e) 機能モジュールがひとつのオブジェクトを同時に利用することが起こりうるが、その際の排他制御を行う方法を提供する。オブジェクトとオブジェクトの間には、包含関係がある(例えば関係型データベースのテーブルとレコードとカラムは、順に包含関係にある)ため、このような複数のオブジェクトを適切に排他制御するためにオブジェクト間の包含関係を考慮した排他制御を実現したモジュール(ロックモジュール)を20 ストレージが提供する。

【0052】ロックモジュールをコンピュータからストレージにダウンロード可能とすることで、排他制御の方法をオブジェクト毎や応用プログラム毎の拡張を可能とする。

【0053】(f) モジュールを高速に実行するために30 ストレージまたは管理マシンにコンパイラを備える。

【0054】(g) 本発明では、複数のストレージへのモジュールのダウンロードを司る管理マシンを置き、必要に応じてモジュールの加工(代表的にはコンパイル)を行った後ストレージにダウンロードする。

【0055】

【発明の実施の形態】本発明の実施の一形態を、図面を参照しながら説明する。なお簡単のため、以下に述べる発明の実施の形態を単に「本実施例」と呼ぶ。

【0056】全体構成図1を用いて、本実施例の全体構成を説明する。40

【0057】図1の全体101は、本実施例が好適に用いられるコンピュータシステムであり、ネットワーク103と、ネットワーク103によって相互接続された1つ以上のコンピュータ102、102'、…および1つ以上のアクティブネットワークストレージ(ANS)104からなる。

【0058】ネットワーク103は、ある団体(企業や学校や類似の団体)の全体や位置部門でよく使用されるLANでもよく、また地理的に分散した複数の地点を結50

合するWANの一部または全部でもよい。またネットワーク103は、計算機間結合網や並列計算機内部のプロセッサ要素間の結合網でもよい。また特殊な場合として、ネットワーク103がコンピュータとANS104をつなぐ入出力用信号線(SCSIケーブル等)でも差し支えない。

【0059】コンピュータ102、102'、…は、いわゆるパーソナル・コンピュータ、ワークステーション、並列計算機、大型計算機、小型携帯型コンピュータ等、任意のコンピュータでよい。

【0060】なお、図1に示したコンピュータ102、102'、…、ネットワーク103、ANS104の数と構成は、例として示したもので、本発明の範囲を限定するものではない。

【0061】ANS104は、拡張型の二次記憶装置(ストレージ)である。ANS104は1つ以上のディスク111とアクティブネットワークストレージコントローラ(ANSC)105とからなる。

【0062】ディスク111は電源断後もデータを保持することが可能な記憶媒体(二次記憶)である。ディスク111のデータ格納単位には、セクタ、トラック等さまざまな呼称があるが、本実施例では一括してブロックと記す。ディスク111は複数のブロックからなり、ブロック単位の入出力を行うことができる。二次記憶がハードディスクであれば、多くの場合ブロックは固定長で512バイトないし4Kバイトである。メインフレーム計算機用のハードディスクであれば、ブロックは固定長の場合と可変長の場合がある。また、テープドライブ等の二次記憶もブロックはその装置毎に決まっている。なお、ブロック単位でなくバイト単位の出入力インタフェースを提供する二次記憶もあるが、ブロックを1バイトと考えることによって本発明を適用することができる。

【0063】ANSC105はANS104の制御を行う部分である。ANSC105はさらにネットワーク制御部106、モジュール実行部107、バッファ管理部108、バッファメモリ109、ディスク制御部110からなる。

【0064】ネットワーク制御部106は、コンピュータ102、102'、…をはじめとする外部からネットワーク103経由で送られてくる入出力要求やその他の通信を受け、また入出力要求への応答やその他の通信をネットワーク103へ送り出す。モジュール実行部107は、ANS104が提供する各種機能を実現する部分である。各種機能の詳細については、後で詳しく述べる。バッファ管理部108は、バッファメモリ109の制御を行う部分である。バッファメモリ109は、ディスク111から得たデータを一時的にためておくメモリである。ディスク制御部110は、ディスク111にブロックの読み出し・書き込みを行わせる制御を行う。ネットワーク制御部106、バッファ管理部108、バッ

ファメモリ109、ディスク制御部110については、従来技術としてよく知られているため、ここではこれ以上詳しく説明しない。

【0065】モジュール実行部107が提供する各種機能は、大きく分けてブロック単位入出力の機能、オブジェクト単位入出力の機能、応用毎の高度機能、その他の機能の4種類に分類される。

【0066】ブロック単位入出力の機能は、従来からストレージがコンピュータに提供している機能である。ブロック読み出し処理（1つ以上のブロックの番号を指定されて、対応するブロックのデータを読み出してコンピュータに返答する処理）と、ブロック書き込み処理（1つ以上のブロック番号と書きこむべきデータを指定されて、対応するブロックにデータを書き込む処理）とが主要な機能である。

【0067】オブジェクト単位入出力の機能は、以下に述べる「オブジェクト」の一部または全部を読み出し・書き込みする処理が主要な機能である。オブジェクトとは、コンピュータ102、102'、…上で動作する応用プログラムがストレージに保存する意味的なデータの塊を指す。オブジェクトの定義は、応用毎または応用分野毎に異なる。オブジェクトの例としては、データベースシステムのテーブル、レコード、カラム等のデータ、ファイルシステムではファイル、また、ファイルシステム上の特定のファイルフォーマットを扱う応用ではその特定のファイルフォーマットのファイル、などがある。

【0068】応用毎の高度機能は、応用プログラムが必要に応じて定義した入出力処理である。例えばソーティング、画像処理、データベースシステムの基本演算、例えば選択処理、写像処理、結合処理、集計処理等が高度機能の例として考えられる。

【0069】その他の機能としては、ディスク111の初期化、バックアップ、複製等の機能が考えられる。

【0070】モジュール実行部107は、上記機能を1つ以上のプログラム・モジュールであるモジュール112、112'、112"、…の組み合わせによって実現する。モジュール112、112'、112"、…は、提供する機能によって機能モジュール113、オブジェクトアクセスモジュール114、ブロックアクセスモジュール115、プロテクションモジュール116、ロックモジュール117の5種類に分類される。さらに新たな種類のモジュールを定義しても差し支えない。

【0071】機能モジュール113は、上記応用毎の高度機能を実現するモジュールである。高度機能の種類によって複数存在しうる。オブジェクトアクセスモジュール114は、上記オブジェクト単位入出力を実現するモジュールである。オブジェクトの種類によって複数存在しうる。ブロックアクセスモジュール115は、上記ブロック単位入出力を実現するモジュールである。ブロックの種類によって複数存在しうる。プロテクションモジ

ジュール116は、モジュール間の呼び出しを選択的に許可または禁止することによって、保護を実現するモジュールである。保護の方法によって複数存在しうる。ロックモジュール117は、オブジェクト単位の排他制御を実現するモジュールである。排他制御の方法によって複数存在しうる。

【0072】ANSインタフェース118は、ANS104とコンピュータ102、102'、…をはじめとする外部とのインタフェースである。応用毎の高度機能、オブジェクト単位入出力、ブロック単位入出力、保護を行うため、モジュール登録・削除119、メソッド起動120、オブジェクト記述データ登録・削除121、保護ポリシー登録・削除122を含む。

【0073】モジュール登録・削除119は、上記モジュール112、112'、112"、…を外からANS104に登録、またはANS104から削除するインタフェースである。メソッド起動120は、ANS104に登録されたモジュールの特定の機能を起動し、応答を受けるインタフェースである。なおメソッドとは、後述するように、モジュールが提供する手続きである。オブジェクト記述データ登録・削除121は、オブジェクトが二次記憶にどのように格納されているかを記述したデータであるオブジェクト記述データを、ANS104に登録、またはANS104から削除するインタフェースである。オブジェクト記述データに関しては、後で詳しく述べる。保護ポリシー登録・削除122は、プロテクションモジュール116が実現するモジュール間の保護の方法を記述したデータである保護ポリシーを、ANS104に登録、またはANS104から削除するインタフェースである。保護ポリシーに関しては、後で詳しく述べる。

【0074】以上が本実施例の全体構成である。

【0075】図6を用いて、ANS104の内部構成および代表的動作を説明する。

【0076】モジュール登録・削除119、メソッド起動120、オブジェクト記述データ登録・削除121、保護ポリシー登録・削除122のいずれかの要求がネットワークまたは入出力用信号線からANS104に送られると、該要求はネットワーク制御部106が受け取る（601）。

【0077】ネットワーク制御部106は、該要求を、その種類に応じてモジュール実行部107か、バッファ管理部108に送る（602、615）。バッファ管理部108に送られる要求は、メソッド起動120のうち、ブロック単位入出力等、従来からSAD型ストレージが提供している機能である。モジュール実行部107に送られる要求は、これ以外のすべての要求である。なわちモジュール登録・削除119、オブジェクト記述データ登録・削除121、保護ポリシー登録・削除122、及びブロック単位入出力以外のメソッド起動120

10

20

30

40

50

がモジュール実行部107に送られる。

【0078】モジュール実行部107の内部には、各種モジュールの管理・実行を行うモジュール管理部650がある。モジュール登録・削除119はモジュール管理部650によってディスパッチされ、適切なモジュールに配送される。

【0079】応用毎の高度機能に関するメソッド起動要求は機能モジュール651に送られて処理される(603)。オブジェクト単位入出力に関するメソッド起動要求およびオブジェクト記述データ登録・削除121は、オブジェクトアクセスモジュール652に送られて処理される(609)。ブロック単位入出力に関するメソッド起動要求で、バッファ管理部が直接扱えない要求はブロックアクセスモジュール653に送られる(611)。保護ポリシー登録・削除122は、プロテクションモジュール654に送られて処理される(609)。モジュール登録・削除119は、モジュール管理部650が直接処理する。すなわち、モジュールの登録の場合には送られてきたモジュールを新たなモジュールとしてモジュール実行部107内に保存する。また、削除の場合には指定されたモジュールをモジュール実行部107内から削除する。

【0080】機能モジュール651は、オブジェクトアクセスモジュール652が提供するオブジェクト単位入出力の機能を利用し、またロックモジュール655が提供する排他制御の機能を利用して(605)、高度機能を実現する。ロックモジュール655は、さらに、オブジェクトアクセスモジュール652の機能を利用して(608)排他制御を実現する。

【0081】機能モジュール651がオブジェクトアクセスモジュール652の機能を利用する際には、機能モジュール651がオブジェクトアクセスモジュール652のメソッドを起動する。このメソッド起動は、モジュール管理部650によってまずプロテクションモジュール654に送られて起動の可否が判定され(604)、起動可ならオブジェクトアクセスモジュール652にメソッド起動要求が送られる(607)。

【0082】オブジェクトアクセスモジュール652は、ブロックアクセスモジュール653の機能であるブロック単位入出力を利用して(610)、オブジェクト単位入出力を実現する。

【0083】ブロックアクセスモジュール653は、バッファ管理部108にブロック取得要求を送るか(613)、または直接バッファメモリ109を参照・変更することによって(614)、ブロック単位入出力を実現する。バッファ管理部108はまた、モジュール管理部650がモジュールを保存する際等に直接起動されうる(612)。

【0084】バッファ管理部108は、バッファメモリ109の内容を管理する(616)。例えば、バッファ

メモリ109のブロックのリプレースメント、ダーティなブロックのディスク111への書き戻し処理の起動、現在バッファメモリ109に存在しないブロックに対するブロック参照要求をディスク制御部110に転送し(617)、ブロックをディスク制御部110からバッファメモリ109へ転送する(618)の起動を行う。

【0085】ディスク制御部110はディスク111へのブロック単位入出力要求を受け、実際にディスク111を駆動して入出力を行う(619)。

【0086】以上が、ANS104の内部構成および代表的動作である。各モジュールの内部構成および動作、オブジェクト記述データ、保護ポリシーについては、さらに以下に詳しく述べる。

【0087】図7を用いて、各モジュール共通の内部構成および代表的動作を説明する。

【0088】オブジェクトアクセスモジュール114、ブロックアクセスモジュール115、プロテクションモジュール116、ロックモジュール117はいずれも、図7に示す構造を共通に持つ。

【0089】モジュール701には1つ以上の手続き(メソッド702)と、0個以上の変数(属性703)がある。モジュール701の他のモジュールまたは外部からメソッド起動704を受けると、対応するメソッドが開始される。メソッド702は、自モジュールの属性703を参照・変更し、または他モジュールのメソッド起動705によって処理を進める。

【0090】機能モジュールの場合、メソッドとして、応用毎の高度機能を外部および他モジュールに提供する。機能モジュールが用いるメソッドは、オブジェクトアクセスモジュールの各メソッド及びロックモジュールの各メソッドである。

【0091】オブジェクトアクセスモジュールの場合、メソッドとしてオブジェクト(とその部分)を指定してその内容を参照・変更する手続きを、外部及び他モジュールに提供する。

【0092】より具体的に、オブジェクトの部分参照するメソッドとして、getObject(Object)(オブジェクトを指定して該オブジェクトの内容全部を参照する)、getObject(Object, offset, size)(オブジェクトとオブジェクト先頭からのバイト数、対象のバイト数を指定して、指定したデータを参照する)、getObject(Object, tag1, tag2, ...)(オブジェクトと、取り出すべき部分を指定するタグとを指定して、指定したデータを参照する) getObject(Object, i, j, k, ...)(オブジェクトと、取り出すべき部分を指定するインデックスとを指定して、指定したデータを参照する) getNextObject(Object)(オブジェクトの次の部分を参照する)を提供する。なお、タグについては、図10、図1

2、図14を用いて後で説明する。

【0093】また、オブジェクトを変更するメソッドとして、setObject (Object, data) (オブジェクトを指定して該オブジェクトの内容全部を変更する)、setObject (Object, data, offset, size) (オブジェクトとオブジェクト先頭からのバイト数、対象のバイト数を指定して、指定したデータを変更する)、setObject (Object, data, tag1, tag2, ...) (オブジェクトと、変更すべき部分を指定するタグとを指定して、指定したデータを変更する)、setObject (Object, data, i, j, k, ...) (オブジェクトと、変更すべき部分を指定するインデックスとを指定して、指定したデータを変更する) setNextObject (Object, data) (オブジェクトの次の部分を変更する)を提供する。これらのメソッドの実現には、ブロックアクセスモジュールのメソッドを用いる。

【0094】上記の各メソッドに加えて、該オブジェクトアクセスモジュールがオブジェクト記述データを用いる場合、後述のようにオブジェクト記述データを登録・削除するメソッドaddObjectDescription (desc) と deleteObjectDescription (desc) を提供する。

【0095】ブロックアクセスモジュールの場合、メソッドとして、ブロックを参照する手続きとブロックを変更する手続きとを、外部および他モジュールに提供する。より具体的には、getBlock (blockID) (ブロック番号を指定してブロックを参照する) と setBlock (blockID, data) (ブロック番号を指定してブロックを変更する) とを提供する。

【0096】プロテクションモジュールの場合、メソッド起動の仕様 (送信元、送信先、メソッド名) を与えられて該メソッド起動の可否を判断する getProtection (source, dest, method) をモジュール管理部に提供する。これに加えて、該プロテクションモジュールが保護ポリシーを登録・削除できる場合、後述のように保護ポリシー記述データを登録・削除するメソッドaddProtectionPolicy (policy) とdeleteProtectionPolicy (policy) を提供する。

【0097】ロックモジュールの場合、オブジェクトにロックをかけるメソッドlockObject (object, lock_mode)、オブジェクトのロックをはずすメソッドunlockObject (object, lock_mode)、オブジェクト間の包含関係を登録・削除するメソッドとして、addObjectRelationship (object, object) とdeleteObjectRelationship

hip (object, object) を、外部及び他モジュールに提供する。

【0098】次に、図8を用いて、オブジェクトアクセスモジュールの中で、特にオブジェクト記述データを用いるモジュールの内部構成および代表的動作を説明する。

【0099】オブジェクト記述データを用いるオブジェクトアクセスモジュールも、モジュール801内に1つ以上のメソッド802と0個以上の属性803があって、メソッド起動804によってメソッド802が起動され、属性803を参照・変更し、または他モジュールのメソッド起動805によって処理を進める。オブジェクト記述データを用いるオブジェクトアクセスモジュールはさらに、オブジェクト記述データ806をメソッド802内に保持し、オブジェクト記述データ登録・削除807に応じて、オブジェクト記述データ806を登録または削除する。オブジェクト記述データ806の構成及び使用方法については、図10、図12、図14を用いて後で説明する。

【0100】次に、図9を用いて、プロテクションモジュールの内部構成および代表的動作を説明する。

【0101】プロテクションモジュールは、モジュール901内に保護判定部902と保護ポリシー903があって、保護判定メソッド起動904によって保護判定部902が起動され、保護ポリシー903を参照することによってメソッド起動の可否を判定する。プロテクションモジュールはさらに、保護ポリシー登録・削除905に応じて、保護ポリシー903を登録または削除する。

【0102】次に、オブジェクト記述データについて説明する。本実施例のオブジェクト記述データには3種類の異なる形式がある。第1の形式は表形式で、ブロック内、ブロック間のデータの並びが固定である場合 (またはそのようなオブジェクトの部分) による形式である。例えば、ファイルシステムのファイルや、データベースシステムのインデックスなどにはこの形式が有利である。

【0103】第2の形式はパーザ形式で、ブロック内、ブロック間のデータの並びを文脈自由文法で記述する方法である。この形式は、ブロック内、ブロック間のデータの並びが比較的自由度を持っている場合 (またはそのようなオブジェクトの部分) による形式である。例えばデータベースシステムの表や、ファイルシステムのディレクトリなどにはこの形式が有利である。

【0104】第3の形式はパターンマッチ形式で、ブロックまたはオブジェクトの特定部分のデータが特定の値かパターンをあるかどうかによってファイルフォーマットを特定する方法である。この形式は、同じ構造のオブジェクトに異なるフォーマットのデータが格納されている可能性がある場合によい形式である。例えばファイルシステムのファイル中に複数のファイルフォーマットの

画像ファイルが格納されており、ファイルフォーマットに応じて高度機能を切り替えたい場合などにはこの形式が有利である。

【0105】図10を用いて、表形式のオブジェクト記述データの構成を説明する。

【0106】オブジェクト記述データ(表形式)1001は、タグ1002、タイプ1003、オフセット1004、サイズ1005、カウント1006、ブロックタイプ1007の6個の要素からなる表である。この表によって、ブロック内及びブロック間のデータの並びを得る。1つの行がブロック内の意味的な一塊のデータを表す。

【0107】タグ1002は、一塊のデータにつけた名前である。この名前は、このデータ部分にアクセスする際に用いられる。タイプ1003は、該データの型を示す。オフセット1004は、該データがオブジェクトの先頭から何バイト目から格納されているかを示す。サイズ1005は、該データのバイト数を示す。カウント1006は、該データが連続して並ぶ場合の総数を表す。ブロックタイプ1007は、該データが他のブロックを参照するブロック番号である場合に、参照先のブロック番号の形式を指定する。

【0108】図11を用いて、表形式オブジェクト記述データの利用例を説明する。

【0109】この例では、文献6に記載のファイルシステムのファイルを一連のオブジェクトとして記述する。該ファイルシステムのファイルは、5種類の異なる形式のブロックによって構成されている。第1に、UFS_inodeオブジェクト1101は、該ファイルのメタデータを格納するオブジェクトである。di_mode、di_nlink、di_uid、di_gid、di_size、di_addr1、di_addr2、di_addr3、di_addr4、di_gen、di_atime、di_mtime、di_ctimeの各要素からなることを示している。例えばdi_modeに着目すると、型はshort型で、オブジェクトの先頭0バイト目から2バイトを占め、繰返し数は1回である。また、別の例としてdi_addr1に着目すると、型はBLOCK型(すなわち他のブロックを参照するブロック番号)で、オブジェクトの先頭12バイト目から3バイトを占め、繰返し数は10回であり、参照先のブロックはUFS_dataオブジェクト1102である。

【0110】同様に、UFS_dataオブジェクト1102、UFS_indirect1オブジェクト1103、UFS_indirect2オブジェクト1104、UFS_indirect3オブジェクト1105が記述されており、全体で0段ないし3段の間接参照を含むファイルの構造が定義されている。

【0111】図12を用いて、パーザ形式オブジェクト

記述データの構成を説明する。

【0112】オブジェクト記述データ(パーザ形式)1201は、タグ1202、初期化コード1203、コンテキスト1204、コード1205の各要素を持つ表である。この表から、ブロックをパースするパーザを構築し、ブロックをパースすることによりブロック内及びブロック間のデータの並びを得る。

【0113】タグ1202は、一塊のデータにつけた名前である。初期化コード1203は、該データをパースする際にあらかじめ実行すべき命令列である。コンテキスト1204は、パースする文脈を指定する。本実施例の指定方法では、文脈自由文法と若干の拡張を施した記述が可能である。コード1205は、文脈をパースした後に実行するべき命令列である。

【0114】図13を用いて、パーザ形式オブジェクト記述データの利用例を説明する。

【0115】テーブル定義1301は、データベースシステムにおける表定義の例を示している。c_id、c_name、c_addressの3つのカラムを持つcustomerテーブルを定義している。

【0116】文法1302は、テーブル定義1301に対応するcustomerテーブルのオブジェクト記述データである。全部で5行の表であり、表全体をパースするcustomer()、行をパースするrecord()、c_idをパースするc_id()、c_nameをパースするc_name()、c_addressをパースするc_address()がそれぞれの行に定義されている。例えば、record()に着目すると、「1つの行は先頭にレコードサイズをあらわすlong型のデータ、続いてc_id、c_name、c_addressが続く」ことがコンテキスト部に示されている。また、recordをパースし終わった際にfoundObject手続きを呼ぶことによって、行を検出したことを宣言する処理がコード部に示されている。このテーブル定義1301により、customerテーブルの各行、各列を検出することができる。なお、この例では用いなかったが、コード1205中でswitchBlock(blockID)手続きを用いることにより、処理対象ブロックを移動することも可能である。

【0117】図14を用いて、パターンマッチ形式オブジェクト記述データの構成を説明する。

【0118】オブジェクト記述データ(パターンマッチ形式)1401は、ファイルフォーマット1402、タグ1403、パターン1404の要素からなる表である。1つのファイルフォーマット1402に対しタグ1403とパターン1404との組が1つ以上対応する。ファイルフォーマット1402はファイルフォーマットの名前である。タグ1403は他のオブジェクト記述データによって与えられる、オブジェクトの一部をなす一

塊のデータの名前である。パターン1404は、タグ1403で示されたデータとパターンマッチを行うパターンである。

【0119】ファイルフォーマット1402に対応付けられたタグ1403とパターン1404との組がすべてマッチした場合に、オブジェクトをファイルフォーマット1402に示されたファイルフォーマットであると判定する。

【0120】例えば、14の1行目および2行目を見ると、あるオブジェクトのheader1というタグをつけたデータが"#!"であって、header2というタグをつけたデータが"/bin/sh"である場合に、このオブジェクトをshell_script(シェルスクリプト)であると判定することができる。

【0121】図15を用いて、保護ポリシーの構成を説明する。

【0122】保護ポリシーは、プロテクションモジュールが保持するデータで、モジュール間の呼び出しを選択的に許可または禁止するために用いる。保護ポリシー記述データ1501は、送信元1502、送信先1503、メソッド1504、許可・禁止1505からなる。

【0123】送信元1502は、メソッド起動の送信元、送信先1503は、メソッド起動の送信先、メソッド1504は、起動対象のメソッド名、許可・禁止1505は、「許可」または「禁止」である。プロテクションモジュールは、メソッド起動に際して、送信元、送信先、メソッドを保護ポリシー記述データ1501の送信元1502、送信先1503、メソッド1504と照合し、許可・禁止1505の値にしたがって、該メソッド起動を許可するか禁止するかを決定する。

【0124】図16を用いて、表形式のオブジェクト記述データを用いるオブジェクト単位入出力処理の手順を説明する。

【0125】オブジェクト単位入出力処理には、上述のようにいくつかのバリエーションがあるが、その代表として、getObject(object, "tag1", "tag2"...)の操作を表形式のオブジェクト記述データを用いて実現する際の手順を説明する。表形式のオブジェクト記述データを用いて実現されるgetObjectの他のバリエーションも、同様の手順で実現できる。getObject(object, "tag1", "tag2"... , "tagN")は、オブジェクトの先頭ブロックのtag1が指定するブロックを参照し、参照先ブロックのtag2が指定するブロックを参照し、という要領でブロックを渡り歩いていき、最後にtagNに合致するデータ部分を参照するメソッドである。

【0126】ステップ1601からステップ1607はループである。ステップ1601で、最初のタグに合致するオブジェクト記述データの行を検索する。ステップ

1602で第2タグがまだあれば(Y)、ステップ1603に制御を移し、なければステップ1608に制御を移す。ステップ1603で、検索結果行のタイプ1003がBLOCK型かを検査する。結果が真(Y)なら、ステップ1604へ制御を移し、偽(N)なら異常終了する。ステップ1604で、処理対象ブロック(ループの最初の場合にはオブジェクトの先頭ブロック)のオフセット1004からサイズ1005分のデータを参照する。該データを参照には、必要に応じてブロックアクセスモジュールの提供するブロック単位入出力を用いる。ステップ1605で、ステップ1604で得たデータを次の処理対象ブロックのブロック番号として保持する。ステップ1606で、使用するオブジェクト記述データを、ブロックタイプ1007用のオブジェクト記述データに切り替える。ステップ1607で、第1タグを捨て、第2タグ以降を第1タグから始まるように左シフトする。ステップ1607の後は、ステップ1601に戻る。ステップ1608では、処理対象ブロックのオフセット1004からサイズ1005分のデータを参照し、結果をgetObject()の返回值として呼び出し元に返す。該データを参照には、必要に応じてブロックアクセスモジュールの提供するブロック単位入出力を用いる。

【0127】以上が表形式のオブジェクト記述データを用いるオブジェクト単位入出力処理の流れである。

【0128】図17を用いて、パーザ形式のオブジェクト記述データを用いるオブジェクト単位入出力処理の手順を説明する。この説明では、getObject(object, i, j, k, ...)の手順を説明する。

【0129】ステップ1701で、引数列i, j, k, ...をターゲット列として保持する。ステップ1702で、オブジェクトのパーザを起動し、処理対象ブロック(最初の場合オブジェクトの先頭ブロック、パーザ処理の進行によって他のブロックに処理対象が移動する場合もある)。パーザはターゲット列を検出するか、オブジェクトの最後までパーザが終了すると、ステップ1702を完了する。ステップ1703で、パーザがターゲット列を検出してパーザが成功したかを検査する。真(Y)なら、ステップ1704でパーザ結果をgetObject()の返回值として呼び出し元に返す。偽(N)なら、異常終了する。

【0130】以上がパーザ形式のオブジェクト記述データを用いるgetObject(object, i, j, k, ...)処理の流れである。

【0131】図18を用いて、パーザ形式のオブジェクト記述データを用いるオブジェクト単位入出力処理の手順を説明する。この説明では、getObject(object, "tag")の手順を説明する。

【0132】ステップ1801で、tagをターゲットタグとして保持する。ステップ1802で、オブジェク

トのパーザを起動し、処理対象ブロック（最初の場合オブジェクトの先頭ブロック、パーザ処理の進行によって他のブロックに処理対象が移動する場合もある）。パーザはターゲット列を検出するか、オブジェクトの最後までパーザが終了すると、ステップ1802を完了する。ステップ1803で、パーザがターゲット列を検出してパーザが成功したかを検査する。結果が真（Y）なら、ステップ1804でパーザ結果を `getObject()` の返り値として呼び出し元に返す。結果が偽（N）なら、異常終了する。

【0133】以上がパーザ形式のオブジェクト記述データを用いる `getObject(object, "tag")` 処理の流れである。

【0134】図19を用いて、パーザ形式のオブジェクト記述データ中でオブジェクトの部分のパーザに成功した際に起動される `foundObject("tag", offset, size, i, j, k, ...)` の処理手順を説明する。`foundObject()` と、図17の `getObject()` または図18の `getObject()` との組み合わせにより、パーザ形式のオブジェクト記述データによるオブジェクト単位入出力が実現される。

【0135】ステップ1901で、`getObject()` が保持するターゲットタグと引数の `tag` とが合致するかを検査する。結果が真（Y）なら、ステップ1903に制御を移し、結果が偽（N）ならステップ1902に制御を移す。ステップ1902で、`getObject()` が保持するターゲット列に、引数 `i, j, k, ...` が合致するかを検査する。結果が真（Y）なら、ステップ1903に制御を移し、結果が偽（N）なら何もせず終了する。ステップ1903で、`getObject()` の処理対象ブロックの `offset` から `size` 分のデータを参照し、パーザ結果として保持する。該データを参照には、必要に応じてブロックアクセスモジュールの提供するブロック単位入出力を用いる。

【0136】以上がパーザ内で用いられる `foundObject` 処理の流れである。以上のように、オブジェクト単位入出力、高機能入出力が、ANSにダウンロードされるモジュールの組み合わせによって実現できる。オブジェクト単位入出力には、オブジェクトの構造を宣言的に記述したオブジェクト記述データが利用できる。これらにより、モジュールの開発コストを低く押さえることができる。

【0137】図20を用いて、ANS利用コンピュータからANSへのモジュール登録動作の構成を説明する。

【0138】ANS利用マシン2001は、ネットワーク2003を通じてモジュール2006をANS 2002に登録する。ANS 2002は前述のように、ANS 2004とディスク2005とからなる。この場合モジュール2006は、ANS 2002の機種に左

右されないために、プラットフォーム独立な言語で記述されるのが、必須ではないが望ましい。プラットフォーム独立な言語としては、インタプリタ型言語をはじめとしていくつかの選択肢がありうる。特に、ネットワーク上での移動を意識して、プラットフォーム独立な実行プログラム、安全なシステム、及び安全な実行時環境を提供する言語が知られており、モジュール2006の記述に好適である。

【0139】しかし、このようなネットワーク上での移動を意識した言語はしばしば、実行時性能が通常のコンパイル型言語に比べて劣る恐れがある。この問題の解決には、2つの解決策が考えられる。第1の選択肢は、ANS内部にコンパイラを持ち、ネットワーク上での移動を意識した言語をより高速な実行プログラム、例えばマシン語で記述された実行プログラムにコンパイルしなおす、という選択肢である。この場合、コンパイルはモジュール管理部650で行うことができる。

【0140】また、第2の選択肢は、ANSの外部に、複数機種のANS向けのコンパイラを持つコンピュータを用意し、このコンピュータ（ANS管理コンピュータ）経由でモジュールをANSへ登録する、という選択肢である。この選択肢は、価格性能比等の理由から、ANSにコンパイラのような大規模ソフトウェアを搭載するのが困難である場合や、ANSがネットワーク上に多数存在する場合に有効である。

【0141】図21を用いて、ANS利用コンピュータからANSへの、ANS管理コンピュータを経由したモジュール登録動作の構成を説明する。

【0142】ANS利用コンピュータ2101は、ネットワーク2103を通じてプラットフォーム独立モジュール2107をANS管理コンピュータ2104に送る。ANS管理コンピュータ2104のANS管理部2110は、プラットフォーム独立モジュール2107の送付先ANSに応じて後述するANS管理表2111を参照し、モジュールコンパイラ2109、2109'、...の中から使用すべきコンパイラを選択する。そして、プラットフォーム独立モジュール2107を該コンパイラでコンパイルし、コンパイル済みモジュール2108を得る。さらにANS管理部2110は、ANS管理表2111によってコンパイル済みモジュール2108の送付先ANSのネットワークアドレスを得、コンパイル済みモジュール2108を該送付先ANSに送付する。送付先ANS 2102は、前述のとおりANS 2105とディスク2106とからなる。

【0143】図22を用いて、ANS管理表の構成を説明する。

【0144】ANS管理テーブル2201は、ANS名2202、ネットワークアドレス2203、機種2204、コンパイラ2205の各要素からなる。複数のANSのネットワークアドレス、機種、モジュールコンパイ

ラの対応関係を保持する。1行が1つのANSに対応する。

【0145】ANS名2202は、ANSの名前である。ネットワークアドレス2203は、該ANSのネットワークアドレスである。機種2204は、該ANSの機種である。コンパイラ2205は、該ANSのモジュールをコンパイルするために使用するモジュールコンパイラの名前である。

【0146】以上がANS管理表の構成の説明である。

【0147】ANS管理部2110、ANS管理表2111、モジュールコンパイラ2109、2109'、…を持つANS管理コンピュータ2104によって、モジュールを高速実行し、かつコンパイルに関する管理コストの低いANSが実現できる。

【0148】ANSの応用例として、ANS利用コンピュータから複数のANSヘストライプ入出力を実現する方法を、図23を用いて説明する。

【0149】ANS利用コンピュータ2301がファイルへの入出力を行う場合、該ファイルを複数のディスクに存在させておくストライピングを行うことによつて、複数のディスク入出力を複数のディスクにばらまくことができ、高速な入出力が可能となる。これを以下の手順で実現する。

【0150】ANS利用コンピュータ2301が複数のANS2304にストライプされたファイルを読み書きする場合、ネットワーク2302を通じてストライプ管理ANS2303ヘストライプ構成情報取得要求2305を送る。この際、ストライプ管理ANS2303は、該ファイルがANS2304にどのようにストライプされているかに関する情報（ファイルのどのブロックがどのANSに格納されているかを示した情報）であるストライプ構成情報を返答する。この機能は、ストライプ管理ANS2303のオブジェクトアクセスモジュール114として実現される。

【0151】ANS利用コンピュータ2301は次に、該ストライプ構成情報から、該ファイルのうち読み書きを行いたい部分を格納しているANSを計算し、該当する1つ以上のANSに対し、入出力要求2306を送る。この2段階の操作により、複数のANSに対するストライプ入出力が実現される。

【0152】ストライプは従来サーバコンピュータが実現し、ストレージはサーバからの入出力要求を受けるだけの受け身の存在であったが、上記の構成方法により、ストレージ自身がストライプを実現することができる。これは、ストレージの構成変更容易性または拡張容易性につながる。例えば、1台のストレージを購入したユーザが、ストレージのボトルネックを解消するために1台のストレージを追加購入し、ストライプ入出力を実現する場合、従来のサーバによるストライプの実現方法では、同時にサーバも新たに購入する必要があり、ストレ

ージの構成変更にも多大なコストがかかっていた。しかし、上記の方法によれば、追加のサーバは必要なく、単に2台のストレージを用意してオブジェクトアクセスモジュール114をダウンロードすれば、ストライプが実現できるという利点がある。

【0153】

【発明の効果】ネットワークに直接結合し、または高機能を提供するストレージを実現するため、(a)応用が利用するデータの塊（オブジェクト）を入出力する機能であるオブジェクトアクセスモジュールを、複数の応用向けの高度機能の共通部分としてストレージが提供し、

(b)オブジェクトが二次記憶に格納されている方法を宣言的に記述するオブジェクト記述データを提供し、

(c)機能モジュールとオブジェクトアクセスモジュールを分離することにより、共通部分であるオブジェクト入出力機能を用いて複数の応用向けの高度機能を実現することを可能にし、(d)機能モジュールからのオブジェクトアクセスモジュールの利用を保護するモジュール（プロテクションモジュール）を提供し、(e)複数のオブジェクトを適切に排他制御するために、オブジェクト間の包含関係を考慮した排他制御を実現したロックモジュールを提供し、(f)ストレージまたは管理マシンにコンパイラを備えることで、モジュールを高速に実行する機構を提供する、これら6点によりモジュールの開発コストを低く抑えた。また(g)多数のストレージが存在するときに、モジュールを配布するために管理コンピュータを用いることにより、ANSを多数利用する際の総所有コストを低く抑えた。

【図面の簡単な説明】

【図1】本実施例の全体構成を示すブロック図。

【図2】サーバ接続型ストレージの構成図。

【図3】ネットワーク接続型ストレージの構成図。

【図4】高機能サーバ接続型ストレージの構成図。

【図5】高機能ネットワーク接続型ストレージの構成図。

【図6】アクティブネットワークストレージの内部構成図。

【図7】モジュールの内部構成図。

【図8】オブジェクト記述データを用いるオブジェクトアクセスモジュールの内部構成図。

【図9】プロテクションモジュールの内部構成図。

【図10】オブジェクト記述データ（表形式）の構成図。

【図11】オブジェクト記述データ（表形式）の使用例を示す図。

【図12】オブジェクト記述データ（パーザ形式）の構成図。

【図13】オブジェクト記述データ（パーザ形式）の使用例を示す図。

【図14】オブジェクト記述データ（パターンマッチ形

式)の構成図。

【図15】保護ポリシーの構成図。

【図16】表形式のオブジェクト記述データを用いるオブジェクト単位入出力処理手順のフローチャート。

【図17】パーザ形式のオブジェクト記述データを用いるオブジェクト単位入出力処理手順のフローチャート(1)。

【図18】パーザ形式のオブジェクト記述データを用いるオブジェクト単位入出力処理手順のフローチャート(2)。

【図19】パーザ内で用いられるfoundObject処理手順のフローチャート。

【図20】ANS利用コンピュータからANSへのモジュール登録動作の構成図。

【図21】ANS利用コンピュータからANSへの、ANS管理コンピュータを経由したモジュール登録動作の構成図。

【図22】ANS管理表の構成図。

10

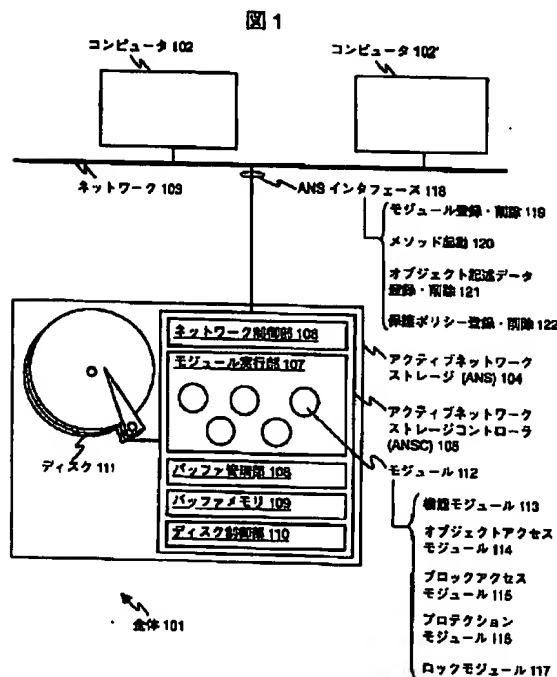
*

*【図23】ANSによるストライプ入出力の構成図

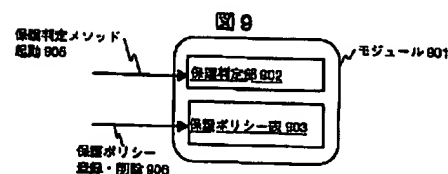
【符号の説明】

101:全体
102、102'、…:コンピュータ、103:ネットワーク、104:アクティブネットワークストレージ(ANS)、105:ANSコントローラ(ANSC)、106:ネットワーク制御部、107:モジュール実行部、108:バッファ管理部、109:バッファメモリ、110:ディスク制御部、111:ディスク、112:112'、112"、…:モジュール、113:113:機能モジュール、114:114:オブジェクトアクセスモジュール、115:ブロックアクセスモジュール、116:プロテクションモジュール、117:ロックモジュール、118:ANSインタフェース、119:モジュール登録・削除
120:メソッド起動
121:オブジェクト記述データ登録・削除
122:保護ポリシー登録・削除。

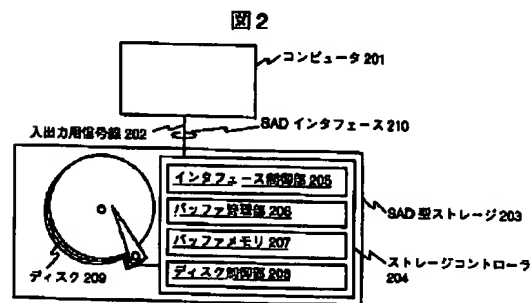
【図1】



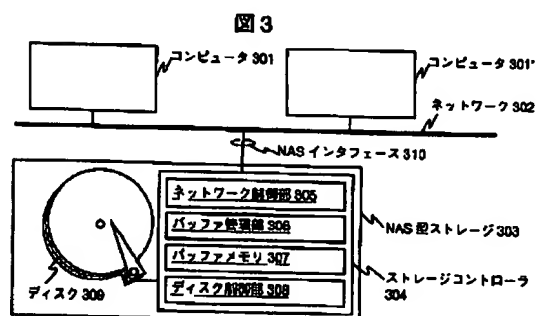
【図9】



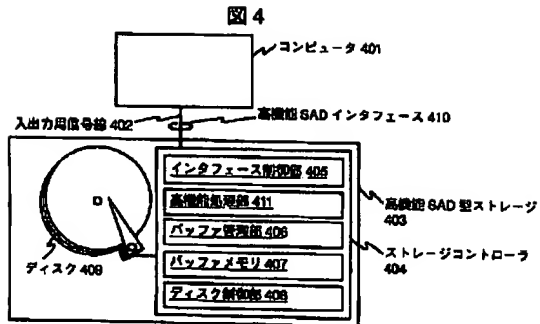
【図2】



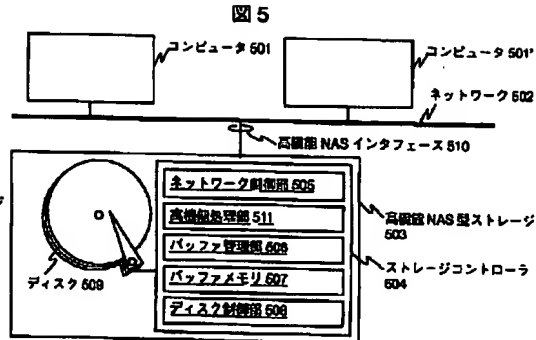
【図3】



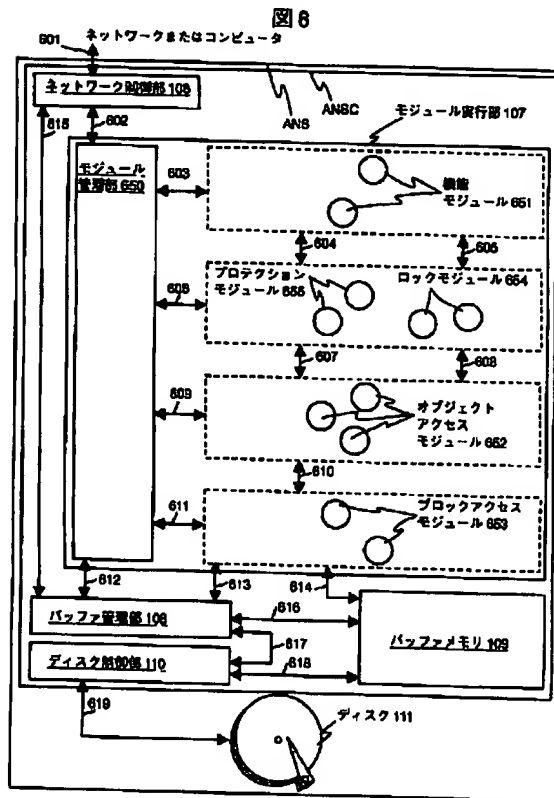
【図4】



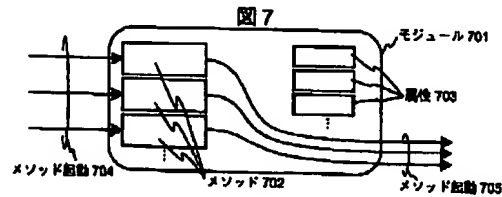
【図5】



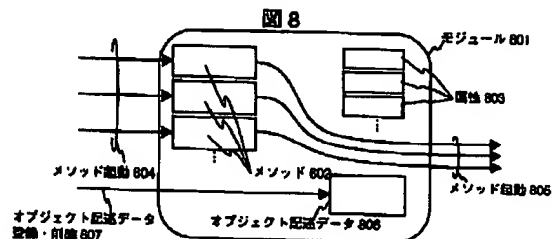
【図6】



【図7】



【図8】



【図10】

図10

オブジェクト記述データ (図形式) 1001

タグ1002	タイプ1003	オフセット1004	サイズ1005	カウント1006	ブロックタイプ1007

【図12】

図12

オブジェクト記述データ (バーザ形式) 1201

タグ1202	初期化コード1203
コンテキスト1204	コード1205

【図11】

図 11

UFS_inode オブジェクト 1101h

タグ1002	タイプ1003	オフセット 1004	サイズ1005	カウント 1006	ブロック タイプ1007
ci_mode	short	0	2	1	null
ci_nlink	short	2	2	1	null
ci_uid	short	4	2	1	null
ci_gid	short	6	2	1	null
ci_size	long	8	4	1	null
ci_addr1	BLOCK	12	3	1	UFS_data
ci_addr2	BLOCK	42	3	1	UFS_indirect1
ci_addr3	BLOCK	45	3	1	UFS_indirect2
ci_addr4	BLOCK	48	3	1	UFS_indirect3
ci_gen	byte	51	1	1	null
ci_mtime	long	52	4	1	null
ci_mtime	long	56	4	1	null
ci_ctime	long	60	4	1	null

UFS_data オブジェクト 1102h

タグ1002	タイプ1003	オフセット 1004	サイズ1005	カウント 1006	ブロック タイプ1007
data	byte	0	1	4096	null

UFS_indirect1 オブジェクト 1103h

タグ1002	タイプ1003	オフセット 1004	サイズ1005	カウント 1006	ブロック タイプ1007
bld	BLOCK	0	4	1024	UFS_data

UFS_indirect2 オブジェクト 1104h

タグ1002	タイプ1003	オフセット 1004	サイズ1005	カウント 1006	ブロック タイプ1007
bld	BLOCK	0	4	1024	UFS_indirect1

UFS_indirect3 オブジェクト 1105h

タグ1002	タイプ1003	オフセット 1004	サイズ1005	カウント 1006	ブロック タイプ1007
bld	BLOCK	0	4	1024	UFS_indirect2

【図13】

図 13

```

create table customer (
  c_id          integer,
  c_name       char(40),
  c_address    varchar(100),
  );
  
```

1301

132

タグ1202	参照コード1203
コンテキスト1204	コード1205
customer()	{ record = 0; column = 0; }
{ record() }	{ null }
record()	{ !col_saved_offset - offset; }
record_size = <long>	{ record++; column = 0; }
c_id() c_name()	{ foundObject("record", record_size, saved_off, record); }
c_id()	{ null }
value = <long>	{ column++; foundObject("c_id", value, sizeof(long), record, column); }
c_name()	{ null }
value = <char>(40)	{ column++; foundObject("c_name", value, sizeof(char)*4, record, column); }
c_address()	{ null }
column_size = <long>	{ column++; }
value = <char>(column_size)	{ foundObject("c_address", value, sizeof(char)*4, record, column); }

【図15】

図 15

解読ポリシー関連データ 1501h

識別元1502	識別元1503	ポリシー1504	許可・禁止 1505
selectionModule	recordAccessModule	getObject("column")	許可
selectionModule	blockAccessModule	getBlock()	禁止
		!	

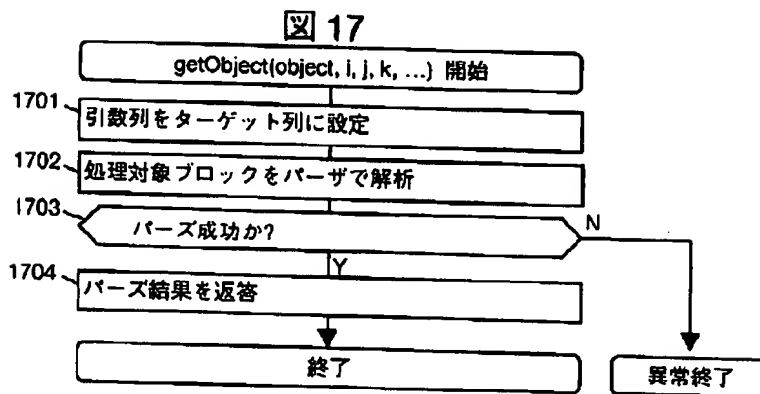
【図14】

図 14

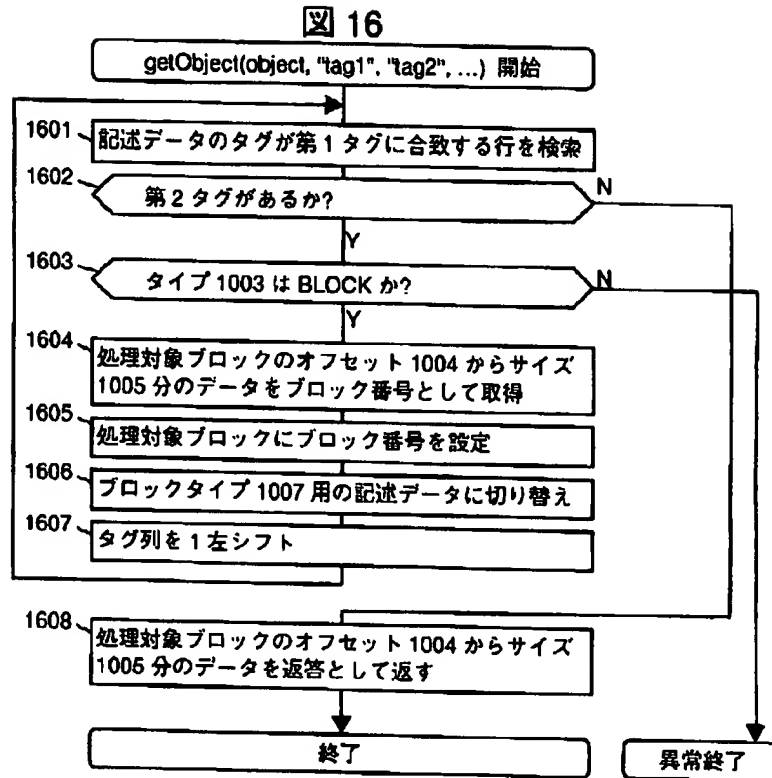
オブジェクトID関連データ (パターンマッチ形式) 1401h

ファイルフォーマット 1402	タグ 1403	パターン 1404
shell_script	header1	"#!"
	header2	"#!nsh"
UNIX executable	header1	ZMAGIC NMAGIC OMAGIC

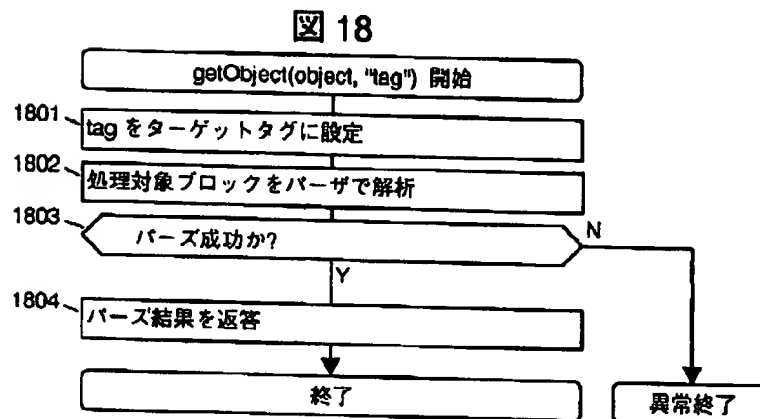
【図17】



【図16】

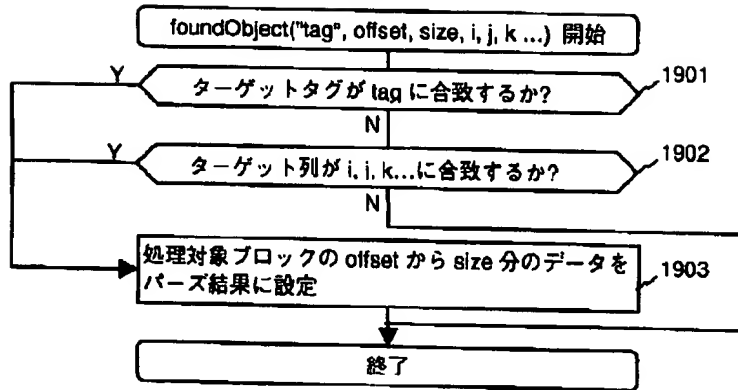


【図18】

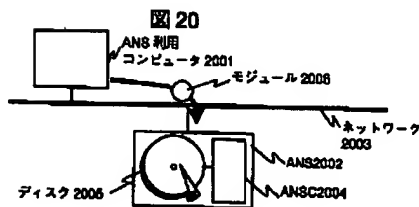


【図19】

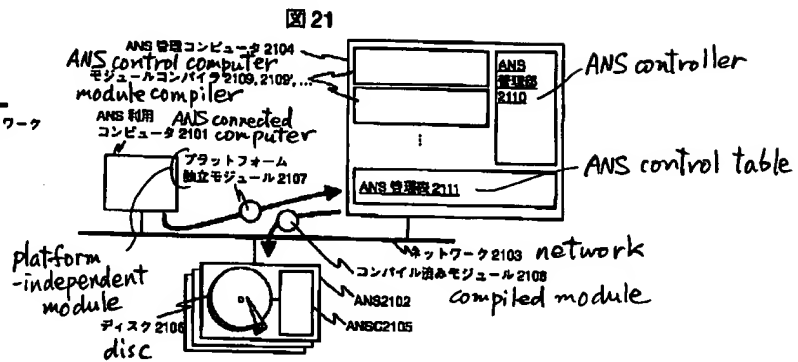
図 19



【図20】



【図21】



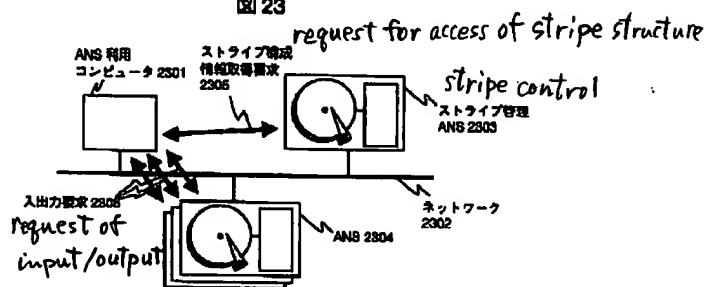
【図22】

図 22

ANS 管理表 2201			
ANS 名 2202	ネットワーク アドレス 2203	機種 2204	コンパイラ 2205
disk1	128.128.11.1	A 社製 342 型	コンパイラ A
disk2	128.128.11.2	B 社製 F01 型	コンパイラ B
		⋮	

【図23】

図 23



フロントページの続き

(72) 発明者 渡邊 直企
東京都分寺市東恋ヶ窪一丁目280番地
株式会社日立製作所中央研究所内

(72) 発明者 富田 亜紀
東京都分寺市東恋ヶ窪一丁目280番地
株式会社日立製作所中央研究所内

(72) 発明者 マシエル フレデリコ
東京都国分寺市東恋ヶ窪一丁目280番地
株式会社日立製作所中央研究所内
(72) 発明者 小田原 宏明
東京都国分寺市東恋ヶ窪一丁目280番地
株式会社日立製作所中央研究所内

(72) 発明者 佐川 暢俊
東京都国分寺市東恋ヶ窪一丁目280番地
株式会社日立製作所中央研究所内
(72) 発明者 杉江 衛
東京都国分寺市東恋ヶ窪一丁目280番地
株式会社日立製作所中央研究所内
F ターム (参考) 5B045 DD15
5B065 BA01 CA02 CA13 CA30 CC03
CE01 CE11 CH18 ZA13
5B082 FA16 FA17